ANALYZING WHEN A LEARNED SUBSET OF OBJECTS EXPLAINS A ROBOT POLICY: DISCOVERY, REMOVAL, CONSEQUENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Object-centric robot policies promise to generalize across scenes and to reveal which objects a robot's decisions rest on. We study In-Loop Subset Explanations (ILSE), policies that select a subset of objects inside the model, train the selection end-to-end and present it as the explanation, and ask when this explanation is earned. We state three conditions for increasingly strong readings of the kept set: The selection must be discovered by the model rather than imposed (Discovery), excluded objects must be absent from the computation (Removal), and it must matter beyond a random subset of the same size (Consequence). We give a test for each condition and apply them to graph-based object-centric policies on three manipulation benchmarks. Excluded objects still reach the action through standard operators, and correcting the removal lowers success by up to 24% relative on CALVIN, mostly through the reductions over excluded objects, and on one split turns an advantage over a pixel policy into a deficit. No tested design point satisfies all three conditions: A labeled selector beats random selection at a matched count, but its choice is imposed by annotations, while a label-free selector that meets Removal and Discovery falls below random selection on average.

1 INTRODUCTION

In robot imitation learning, object-centric representations have become a popular alternative to full image embeddings. Rather than mapping pixels directly to actions, many manipulation policies first decompose the scene into objects and condition on that decomposition, whether as object-level features (Devin et al., 2018; Zhu et al., 2023b), slots learned without supervision (Locatello et al., 2020; Chapin et al., 2025), object-centric tokens (Bendikas et al., 2025), or a graph over detected objects (Qi et al., 2025; Qian et al., 2026). Because the input is an explicit set of objects, a model that selects a subset of them inside its forward pass and acts on that subset alone appears to state which objects its decision depended on, so the retained subset can be read as its explanation. Recent object-centric policies make exactly this claim and present the retained objects as an explanation of the policy and of what the benchmark requires, while reporting no loss in performance compared to pixel-based policies (Mattes et al., 2026), and related approaches read the most attended object as the reason for a decision (Renz et al., 2023).

We ask when this claim is earned, and whether it can be earned without a loss in performance. We call a method an *In-Loop Subset Explanation (ILSE)* if it *(i)* selects a subset of objects inside the model, *(ii)* trains that selection end-to-end with the model, and *(iii)* presents the retained subset as the explanation (Figure 1). Such methods also exist outside robotics, where self-explaining graph networks (Miao et al., 2022; Serra & Niepert, 2024) and rationale models (Lei et al., 2016) select part of their input, predict from it, and offer it as the interpretation of their prediction (Table 1). Unlike post-hoc explanations, which can be checked by re-running a fixed model on the retained input (Ying et al., 2019; Covert et al., 2021), an ILSE never builds the retained subset as a standalone input, and its jointly trained selector can signal the output through *which* objects it keeps rather than what they contain (Jethani et al., 2021; Puli et al., 2024). Closed-loop control makes both problems harder, because removing objects from a trained policy moves the robot into unfamiliar states whose errors compound over the episode (Ross et al., 2011), so a drop in success mixes the effect of the removed objects with that of these states (Hooker et al., 2019; Hase et al., 2021).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: (a) An In-Loop Subset Explanation selects objects inside the policy, trains the selection end-to-end and presents the kept subset as the explanation, which raises one question per condition. (b) On CALVIN D→D, no tested design point satisfies all three conditions: The corrected labeled selector satisfies Removal and Consequence but copies the annotation list, and the label-free selector satisfies Discovery and Removal but falls below random selection. Bars show the average number of completed instructions.

We state three conditions for increasingly strong readings of the kept set and detail them in Section 3. The *Discovery-Condition (D-C)* requires that the selection is the model's own decision rather than imposed by the object annotation list it is judged against, so that the kept set shows what the model found. The *Removal-Condition (R-C)* requires that excluded objects are truly absent from the computation, so that the action depends only on the kept set. The *Consequence-Condition (C-C)* requires that the selection matters beyond a random selection of the same size, so that what the model found is informative. A labeled selector can thus satisfy R-C while failing D-C, so that its kept set restates the annotation instead of revealing what the task requires. The conditions must be tested separately, and our tests take the selection as given, so they do not rule out that the choice of objects itself carries information about the excluded ones.

We test the three conditions on RoboCasa (Nasiriany et al., 2024), CALVIN (Mees et al., 2022) and RoboTwin 2.0 (Chen et al., 2025), and compare every design point against a pixel-based policy to measure what a leak-free explanation costs. Our main policies are graph-based object-centric policies with a selector trained with or without an object annotation list and a removal that leaks or is corrected (Mattes et al., 2026; Yu et al., 2022). We present two label-free design points built to satisfy D-C and R-C together and include policies without a graph encoder in Appendix A.2. We also apply the removal test to three external methods chosen in advance: Top-k graph pooling (Gao & Ji, 2019; Lee et al., 2019), a self-explaining graph network (Miao et al., 2022), and differentiable token pruning (Rao et al., 2021). Our contributions are the following:

Three conditions for increasingly strong readings of the kept set, Discovery, Removal and Consequence, for when the subset kept by an ILSE explains the model that selected it, with a test for each condition.

Two failures in standard components: Leaks from excluded objects through which they still reach the action, and budgets that fill leftover slots with low-scored objects, which costs success on the corrected path.

Experiments on three manipulation benchmarks showing that no tested design point satisfies all three conditions: Labeled selectors satisfy Removal and Consequence but not Discovery, and label-free selectors that satisfy Discovery and Removal fall below random selection on average.

2 Related Work

We group prior work by which of the three ILSE properties it has (Table 1): *(i)* Object-centric models without selection, *(ii)* selection outside the policy, *(iii)* selection learned end-to-end without an explanation claim, and *(iv)* selection learned end-to-end and presented as the explanation. We restrict ourselves to explanations given as a subset of the objects a policy acts on, because this is the format object-centric policies offer about themselves.

Object-centric policies without selection. Many manipulation policies condition on a decomposition of the scene into objects, using object proposals (Zhu et al., 2023b), object-centric 3D representations (Zhu et al., 2023a), slots (Chapin et al., 2025; Mosbach et al., 2025), or graphs over objects (Lin et al., 2022; Vosylius & Johns, 2025). All objects reach the policy, and claimed interpretability rests on visualized attention (Mosbach et al., 2025; Vosylius & Johns, 2025), whose value

Table 1: Methods that act on a set of objects, grouped by the ILSE properties they have. The conditions bind only in group (iv), and elsewhere the entries show how a selection would fare if read as an explanation. D-C marks independence from the object annotation list used for evaluation, so a selection from independent labels, such as motion, can satisfy it, and whether a selection decides is reported separately in Section 4.2. ✓/× measured, (✓)/(×) expected from the design but not tested, n.d. tested but not detected, ? unknown, – not applicable. C-C gives the consequence test a paper reports or, for our design points, our result.

Method	Selection	Removal	D-C	R-C	C-C
(i) Object-centric, no selection					
VIOLA (Zhu et al., 2023b)	none	soft weights	–	–	–
PlanT (Renz et al., 2023)	none	attention	–	–	expert
(ii) Selection outside the policy					
Scene graphs (Qian et al., 2026; Qi et al., 2025)	VLM or LLM	outside policy	(×)	(✓)	?
PLOI (Silver et al., 2021)	planner labels	outside policy	(×)	(✓)	random
Motion-Grounded Selection (ours)	motion labels, frozen	indicator	✓	✓	n.d.
(iii) Selection learned end-to-end, no explanation claim					
TopKPool, default (Gao & Ji, 2019)	label-free	own-score gate	(✓)	✓	–
TopKPool with minimum score, SAGPool (Gao & Ji, 2019; Lee et al., 2019)	label-free	score-gated	(✓)	×	–
DynamicViT (Rao et al., 2021)	label-free	masked softmax	(✓)	✓	–
LightVLA (Jiang et al., 2025)	label-free	straight-through	(✓)	?	eval-time
TransMASK (Parekh et al., 2026)	label-free	soft mask	(✓)	(×)	–
(iv) ILSE: selection learned end-to-end and presented as the explanation					
SIR (Mattes et al., 2026)	labeled	score-gated	×	×	?
GSAT (Miao et al., 2022)	label-free	soft weights	(✓)	×	?
L2XGNN (Serra & Niepert, 2024)	label-free	hard	(✓)	?	?
Rationale models (Lei et al., 2016)	label-free	hard, sampled	(✓)	?	?
Tang et al. (2020)	label-free	hard top-k	(✓)	(✓)	?
VGIB (Yu et al., 2022)	label-free	noise substitution	(✓)	×	n.d.
Labeled, corrected (ours)	labeled	straight-through	×	✓	✓
Label-free (ours)	label-free	sampled, straight-through	✓	✓	×

as an explanation is contested (Jain & Wallace, 2019; Wiegreffe & Pinter, 2019), or on post-hoc explainers (Lin et al., 2022). PlanT (Renz et al., 2023) reports the most attended object as the reason for a driving decision, but tests it by passing that object to a rule-based expert, which shows whether the object suffices for the expert rather than whether the planner relied on it.

Selection outside the policy. Here an independent module decides which objects the policy sees before it runs, such as a pretrained segmenter (Bendikas et al., 2025), a language or vision-language model (Qian et al., 2026; Qi et al., 2025), or a separately trained network that predicts which objects a planner needs (Silver et al., 2021). Because excluded objects never enter the policy, removal holds by construction, but for the same reason the selection reflects the judgment of the external module, which satisfies D-C only when that module does not use the annotation the selection is judged against.

Selection learned end-to-end without an explanation claim. Several methods learn a selection jointly with the model for efficiency. Top-k graph pooling scales kept nodes by their scores to pass a gradient through the discrete choice (Gao & Ji, 2019; Lee et al., 2019), and learned pooling can match random assignment (Mesquita et al., 2020). DynamicViT (Rao et al., 2021) instead masks the attention denominator, because zeroed tokens would otherwise still act through the softmax. Learned pruning and soft masks have since reached robot policies (Jiang et al., 2025; Parekh et al., 2026), but none of these methods presents its selection as an explanation.

Selection presented as the explanation. Rationale models in language processing select input tokens and predict only from them (Lei et al., 2016), but jointly trained selectors can encode the label in which parts they choose (Yu et al., 2019; Jethani et al., 2021; Puli et al., 2024). Self-explaining graph networks select subgraphs (Yu et al., 2022; Miao et al., 2022; Serra & Niepert, 2024) whose faithfulness is formalized as sufficiency and necessity, yet they can report subgraphs that do not reproduce the prediction (Chen et al., 2024) or that are unrelated to how the model predicts (Azzolin et al., 2026). Closest to our Removal-Condition, Azzolin et al. (2024) identify architectural desiderata, such as hard scores and a readout over the explanation only, which our R-C requirements share and extend to a test against a constructed input in closed loop. Soft concept values in concept bottleneck models similarly leak information beyond the concept they name (Koh et al., 2020; Havasi et al., 2022), and object-centric policies present their retained objects as the explanation (Mattes et al., 2026). Each of these addresses one of our conditions, but none states them jointly or tests them on a closed-loop policy.

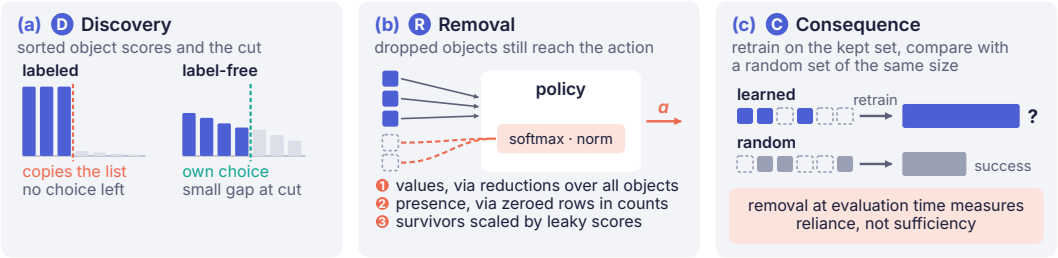


Figure 2: How each condition fails. (a) The labeled selector saturates, so its budget keeps exactly the listed objects, while the label-free selector decides with a small gap at the cut. (b) Excluded objects reach the action through reductions over all objects, zeroed rows that still count, and survivors scaled by leaky scores. (c) Consequence compares policies retrained on the kept and on a random set of the same size, since removal at evaluation time measures reliance, not sufficiency.

Evaluating explanations by removal. Removing inputs from a trained model creates out-of-distribution inputs (Hase et al., 2021), which motivated retraining after removal (Hooker et al., 2019) and in-distribution evaluation for graph explanations (Amara et al., 2023), while agreement with ground-truth masks measures plausibility rather than faithfulness (Faber et al., 2021; Jacovi & Goldberg, 2020). In reinforcement learning, saliency maps have been tested counterfactually (Atrey et al., 2020; Puri et al., 2020) and attention has been built into agents (Mott et al., 2019). We carry these concerns into closed-loop control, where the policy generates its own test distribution, and our permutation test for D-C is a code-level check in the spirit of randomization tests (Adebayo et al., 2018). Causal confusion in imitation learning (De Haan et al., 2019) and random object dropout (Park et al., 2021) suggest why smaller object sets can help, and ERASER formalizes sufficiency and comprehensiveness for rationales (DeYoung et al., 2020).

3 Conditions for an In-Loop Subset Explanation

At time t the policy observes a set of objects $\mathcal{O}_t = \{o_1, \ldots, o_{n_t}\}$ with features x_i and a task instruction c. A selector assigns each object a score $s_i = g_\phi(\mathcal{O}_t, c)_i$, and a selection rule turns the scores into a kept set $S_t \subseteq \mathcal{O}_t$, either the k highest-scoring objects under a budgeted rule or every object with $s_i \geq \tau$ under a threshold rule. The policy acts on the kept set, $a_t = \pi_\theta(S_t, c)$, but in practice S_t is never built as a separate input, and the implementation computes $a_t = f_\theta(\mathcal{O}_t, m_t, c)$ from the full set and a mask $m_t \in \{0,1\}^{n_t}$, training selector and policy jointly. The explanation claim of an ILSE is that S_t is the set of objects on which a_t depends, which R-C secures, and the stronger claims that the model found these objects and that they are informative, which D-C and C-C secure (Figure 2). We present the conditions in the order of the three parts of the definition.

3.1 Discovery-Condition

Definition. The kept set must be the model's decision rather than imposed by the object annotation list it is judged against, and the selection rule must actually discriminate among the objects. Because D-C is judged against the object annotation list, a selection trained from independent labels, such as the motion of objects, can satisfy it.

Failure through identity. A selector trained with a loss on the object annotation list learns to reproduce that list, so its agreement with the list measures plausibility rather than what the policy found (Jacovi & Goldberg, 2020). We check this with a permutation test, a sanity check on the code path: A selector that satisfies D-C produces bit-identical outputs and training loss when the list is replaced by a permuted one, whereas a selector trained on the list does not. The labeled selector fails this test by construction, and the conditions are not nested, although the claims they secure are.

Failure through cardinality. The size of the kept set is part of the selection. If the budget is derived from the object annotation list, for instance as the number of annotated objects per task, a selector trained without any loss on the list is still told how many objects to keep, which is a form

of encoding (Puli et al., 2024). A threshold rule avoids this budget, but the size of its kept set then depends on every object's score, including the excluded ones.

Failure through non-decision. Let $Z_t = \{i : s_i \geq \tau\}$ be the objects the scorer rates confidently. If $|Z_t| \leq k$, top-k keeps all of Z_t regardless of their ranking, a *no-choice* decision, in which the kept set is fixed by the budget rather than by the scores, even if the scorer separates the objects sharply. The leftover $k - |Z_t|$ slots are then filled with low-scored objects, arbitrarily when their scores barely differ, which we measure by the gap between the last kept and the first dropped score, normalized by the range of scores in the scene, and at a gap of zero the implementation rather than the input decides which object survives. A threshold rule fails in the same way when every score clears the cut and the selector keeps everything. More details are given in Appendix A.3.

Requirement. The selector is trained from a signal other than the object annotation list used to judge it and its budget is not derived from that list, which our tables mark as D-C, and its rule decides, with gaps at the cut well above zero, which we report separately as a diagnostic.

3.2 Removal-Condition

Definition. Let $\mathcal{C}(S_t)$ be an input constructed from the retained objects alone, as if the others had never been observed, with its own normalization statistics and its own edges if the objects form a graph. R-C holds at a comparison point p, such as the encoder output or the final action, if

$$f_\theta^{(p)}(\mathcal{O}_t, m_t, c) = f_\theta^{(p)}(\mathcal{C}(S_t), \mathbf{1}, c), \tag{1}$$

where $f_\theta^{(p)}$ denotes the computation up to p. If both sides agree, excluded objects are absent, and the retained objects are processed exactly as if the others had never been observed. The condition is conditional on the selection, since the scorer must read every object, including those it excludes, to decide that they are irrelevant. R-C therefore certifies the computation downstream of the selection, and it does not rule out that the choice itself carries information about the excluded objects, which our D-C test checks only for the annotation list.

Failure through leakage. We write the residual of eq. (1) as $\Delta_t = f_\theta^{(p)}(\mathcal{O}_t, m_t, c) - f_\theta^{(p)}(\mathcal{C}(S_t), \mathbf{1}, c)$. A *value-dependent* leak makes Δ_t change when the features of excluded objects change. A *presence-only* leak makes Δ_t nonzero but constant in those features. Both arise from an aggregation over the pre-removal set, where a reduction runs over all objects instead of S_t. *Survivor distortion* makes Δ_t nonzero although no excluded object enters the computation directly, because retained objects are weighted by a score that depends on the excluded objects. A weight computed from a retained object's own features, as in default top-k pooling (Gao & Ji, 2019), leaves eq. (1) intact, since the constructed input receives the same weight, but changes what the object contributes, which we report separately. Perturbing the excluded objects detects only value-dependent leaks, since the other two classes do not depend on the excluded values, so we test R-C by comparing against the constructed input directly.

Requirements. A design satisfies R-C if *(R1)* selection happens before any computation that mixes objects, *(R2)* every downstream reduction runs over S_t only, and *(R3)* the readout weights retained objects independently of the excluded ones, for example by exactly one instead of by scores computed from all objects. These requirements agree with the architectural desiderata proposed for modular graph networks (Azzolin et al., 2024), which we extend with the test against a constructed input. To remain trainable, a design that satisfies them must pass the gradient to the scorer through a straight-through estimator, hard in the forward pass and soft in the backward pass, since an exact indicator passes none (R4).

3.3 Consequence-Condition

Definition. The selection must matter. A policy trained to act on the kept set must behave differently from one trained without the selection, and the learned selection must outperform a random selection of the same kept fraction trained the same way. This is a claim of *sufficiency*, that the kept objects are enough, together with *selectivity*, that these objects are better than arbitrary ones. When several subsets carry the same information, a faithful selection can fail C-C, so a failure means that the kept set is no more informative than a random one, not that it is unfaithful.

Failure through evaluation and construction. Removing objects at evaluation time from a policy trained on full sets measures reliance, not sufficiency, so sufficiency requires removal at training time (Hooker et al., 2019), on a path that itself satisfies R-C. A random baseline that keeps a different fraction of objects confounds selectivity with set size, treating rollouts rather than random draws as independent units overstates significance, and if a robot-only or blind policy approaches the full policy, the benchmark leaves the selection nothing to matter for.

Requirement. Training-time removal on a path that satisfies R-C, a random baseline at the same kept fraction with draws as units, and benchmark floors from robot-only and blind policies.

3.4 Why the Conditions Pull Against Each Other

Selecting outside the model makes R-C trivial but ties D-C to whatever labels the external module was trained on, while selecting inside the model makes D-C possible and R-C difficult. Training end-to-end adds a second tension, because an exact indicator passes no gradient, and the standard remedy, weighting survivors by scores computed from all objects, violates R3. Straight-through estimation resolves this (Rao et al., 2021), but objects removed from the computation receive no gradient, so a label-free selector can only reconsider an object it dropped if its training mask is sampled, and the variational information bottleneck (Yu et al., 2022) avoids this only by replacing discarded objects with noise, which is not removal. Finally, no budget rule serves all three conditions at once: A budget derived from the annotation list fails D-C, a budget proportional to the number of objects lets the count of kept objects reveal how many were excluded, and a constant budget cannot match the kept fraction of another selector across tasks. Our label-free design point uses a proportional budget with a constant taken from the labeled selector's kept fraction, so it inherits the count channel and one annotation-derived number.

4 Experiments

We follow the protocol of SIR (Mattes et al., 2026), extracting objects and their features only from the benchmark's static cameras, so that the object set is the policy's only view of the scene. On RoboCasa (Nasiriany et al., 2024) we report success over 200 rollouts per task on the 16 kitchen tasks outside pick-and-place, on CALVIN (Mees et al., 2022) the average number of completed instructions in chains of five over 1,000 chains per seed on the D→D and ABC→D splits, and on RoboTwin (Chen et al., 2025) success on four tasks with 50 rollouts each per seed, on clean scenes and with added distractors. We use four seeds on CALVIN and RoboCasa and three on RoboTwin, pair differences across seeds, and train all policies within a benchmark for the same number of optimizer steps, so comparisons are at a matched budget rather than at convergence. All 24 RoboCasa tasks, intervals, standard deviations and further ablations are reported in Appendix A.2.

4.1 Policy and Design Points

Policy. All policies share the MDT architecture (Reuss et al., 2024), without its auxiliary objectives. Following SIR and other object-graph policies (Mattes et al., 2026; Lin et al., 2022; Qian et al., 2026), each frame's objects form a graph whose nodes carry appearance and geometry features and are encoded by a two-layer graph network, with one token per node for the policy. Learned selections score the nodes with a two-layer transformer that reads the features of all objects, conditioned on the instruction, before the graph network runs (R1). On the *leaky path*, reductions still run over all nodes and kept nodes are scaled by their scores, and the *corrected path* restricts reductions to the kept set, weights every kept node by exactly one (R2, R3), and trains the scorer through a straight-through estimator (R4).

Design points. *Labeled, leaky* is SIR as originally implemented (Mattes et al., 2026), and *Labeled* is the same selector on the corrected path. *Label-free, leaky* is the variational information bottleneck (Yu et al., 2022), which sets its own kept fraction but replaces discarded nodes with noise. *Label-free* is built to satisfy D-C and R-C together, on CALVIN and RoboTwin. Its scorer is trained only through the policy loss and a sparsity term toward a target fraction r, set to the labeled selector's kept fraction, with a training mask sampled from the scores so that every object is occasionally kept, and at evaluation it keeps the $\lceil r\, n_t \rceil$ highest-scoring nodes. A select-then-predict design outside ILSE, trained from motion labels, is reported in Appendix A.6.

Table 2: Success of each design point and the conditions it meets, where R-C lists the requirements of Section 3.2. No design point satisfies all three conditions. Random selections keep about as many objects as the selection they are compared with, and keeping every robot link is an exploratory comparison added after the main results. ✓/× measured, n.d. tested but not detected, ? not tested, – not applicable or not run. Details per cell in Appendix A.1 and A.2.

	Conditions			CALVIN (avg. len.)		RoboCasa (%)	RoboTwin (%)	
Design point	D-C	R-C	C-C	D→D	ABC→D	16 tasks	clean	distractors
References								
Pixel policy	–	–	–	1.76	1.40	12.84	46.3	1.5
Blind policy	–	–	–	0.44	–	–	1.5	2.5
Robot-only	–	–	–	1.08	–	19.85	–	–
Unsparsified graph	–	–	–	1.25	1.04	17.37	22.2	0.3
Fixed annotation list	–	–	–	–	–	20.09	–	–
Random, labeled size	–	–	–	0.70	–	19.01	15.8	–
Random, label-free size	–	–	–	0.70	–	–	22.2	1.0
Random, robot links kept	–	–	–	1.12	–	–	–	–
ILSE								
Labeled, leaky (SIR)	×	R1	?	2.12	2.20	25.09	26.7	0.8
Labeled, corrected (ours)	×	R1–R3	✓	1.62	1.71	21.62	25.8	not run
Label-free, leaky (VGIB)	✓	none	n.d.	1.14	1.11	18.55	–	–
Label-free (ours)	✓	R1–R3	×	0.46	–	–	13.0	3.3
Label-free, robot links kept (ours)	✓	R1–R3	n.d.	1.18	–	–	–	–

Baselines. A pixel policy runs the same architecture on camera images through an ImageNet-pretrained ResNet-18 encoder, and the unsparsified graph policy receives every node. A robot-only policy receives only the robot's nodes, a blind policy only proprioception, and a random selection keeps random nodes at the kept fraction of the selector it is compared with (Appendix A.1).

4.2 Discovery

Dependence on the annotation list. Only the labeled selectors depend on the annotation list. With a permuted list, the label-free design points and their random baselines produce bit-identical actions and training loss, while the labeled selector's training loss and kept set change (Appendix A.3). Selectors trained on the list reproduce it almost perfectly, with an AUROC of 0.9997 on CALVIN against 0.70 for the label-free leaky selector (Appendix A.3).

What the labels contribute. The content of the annotation list matters, but selection without it gains little. On RoboCasa, weighting all objects softly reaches 20.64% with the list, 15.30% with a permuted list and 18.21% without a loss on the list, against 17.37% unweighted (Appendix A.3). Without labels, VGIB reaches 1.14 on CALVIN against 1.25 for the unsparsified graph, and the label-free design point 0.46 on CALVIN and 13.0% on RoboTwin, against 1.62 and 25.8% for the labeled selector (Table 2).

Selections and non-decisions. Budgeted selectors trained on the list fill leftover slots with low-scored objects. Their budget keeps every confidently scored object in 94–100% of frames, and where it exceeds them, the remaining slots go to objects rated low, which lowers the labeled selector's success on CALVIN from 1.62 to 1.29 when two such slots are added (Appendix A.3). The threshold of the label-free leaky selector keeps every object in 7–35% of CALVIN frames, depending on the seed, and the label-free design point decides with a small gap at the cut. Its selection is consistent across seeds but unstable in time and keeps other objects than the annotated ones: On CALVIN its kept sets agree across seeds twice as often as chance, yet change between consecutive steps far more often than the labeled selector's (0.18 against 0.01), and it keeps unannotated scene objects more often than annotated ones, even when every robot link is kept (Appendix A.3).

4.3 Removal

Where excluded objects re-enter. Excluded objects re-enter through their values when a reduction runs over all objects, as in the softmax denominators and normalization statistics of the leaky SIR path or in a self-explaining graph network (Miao et al., 2022) that never commits to a subset, through

their presence when a removed object still counts as a zeroed row, and through survivors scaled by scores computed from all objects, as in the leaky SIR path and self-attention graph pooling (Lee et al., 2019), whereas default top-k pooling scales them only by their own gate (Gao & Ji, 2019). In the leaky CALVIN policy, replacing only the features of excluded objects changes the action by about a tenth of its sampling spread and occasionally reverses the gripper command, whereas the corrected policy is unaffected (Appendix A.4). Differentiable token pruning (Rao et al., 2021) and our label-free design points match the constructed input exactly at their output, and our corrected path at the encoder output (Appendix A.4).

What closing them costs. Closing every leak lowers success from 2.12 to 1.62 on CALVIN D→D, from 2.20 to 1.71 on ABC→D, from 25.09% to 21.62% on RoboCasa, and from 26.7% to 25.8% on RoboTwin (Table 2), so the corrected policy falls below the pixel policy on CALVIN D→D. Reversing only R2, so that reductions again run over all objects while survivors keep weight one, recovers 0.44 of this 0.51 drop on CALVIN, so most of the leaky policy's advantage runs through the reductions over excluded objects, although the leak per frame is small (Appendix A.4). Because reversing R2 also changes how many objects each reduction normalizes over, this recovery bounds the value of leaked information from above rather than isolating it. Closing only some leaks can cost more than closing none: On RoboTwin, a first fix that scaled kept objects by their scores reaches 21.5%, below both the leaky and the corrected policy (Appendix A.4).

4.4 Consequence

Where the scene matters. On CALVIN D→D, the selected scene objects add about half a completed instruction over a robot-only policy (1.62 against 1.08), on RoboCasa the robot-only policy already exceeds the unsparsified graph (19.85% against 17.37%), and on RoboTwin a blind policy reaches only 1.5% (Table 2).

Learned against random selection. The C-C verdicts rest mainly on CALVIN, where the contrasts are clearest. The labeled selector beats random selection at a matched count on CALVIN (1.62 against 0.70) and RoboTwin (25.8% against 15.8%), and on RoboCasa its annotation list beats a random list of the same size (20.09% against 19.01%). The label-free design point falls below random selection on CALVIN in every seed (0.46 against 0.70) and on RoboTwin in the three pre-registered seeds (13.0% against 22.2%), where a six-seed extension remains negative only on average (Appendix A.5). Its sampled training mask also makes its kept set far less stable over time than those of the random baselines, which on RoboTwin train on one fixed subset, so part of this gap may reflect training difficulty rather than selection quality. Keeping every robot link lifts it to 1.18, level with a random selection that also keeps them (1.12), but with every robot link kept even the labeled selector does not detectably beat random, so part of its advantage may come from which robot links it keeps.

Distractors and evaluation-time removal. Under distractors every policy trained on clean scenes falls to the level of the blind policy on RoboTwin, the pixel policy included, which a two-by-two design attributes to the training data (Appendix A.5). Closing the leak only at evaluation lowers RoboCasa success from 25.09% to 16.27%, whereas training with true removal gives 21.62%, so evaluation-time removal overstates the cost of removal (Appendix A.5).

5 Discussion

What the conditions separate. No tested design point satisfies all three conditions, and each fails at a different level of the claim. A labeled selector with corrected removal is a faithful explanation of what its action depends on, since it satisfies R-C and beats random selection at a matched count, but its kept set restates the annotation list rather than showing what the model found, and with every robot link kept even its advantage over random selection is no longer detectable. The label-free design points show that D-C and R-C can be met together, but they fall below random selection, mainly because they drop robot links, and with every robot link kept they reach the random level while still avoiding the annotated objects. A policy loss passed through a straight-through estimator is thus a weak discovery signal in our setting, which does not show that the conditions are incompatible, since other estimators or task-related signals such as object motion may do better.

Explanation and performance. SIR reports that selecting objects explains a policy without costing performance relative to pixel-based policies (Mattes et al., 2026). This holds for the leaky imple-

mentation, while closing the leaks costs nearly a quarter of performance on CALVIN, a seventh on RoboCasa and little on RoboTwin, and turns the advantage over the pixel policy into a deficit on CALVIN's in-distribution split. Two effects contribute: The object representation already trails the pixel policy without any selection on CALVIN and RoboTwin, and on CALVIN most of the cost of correction comes from the reductions over excluded objects, since reversing only the reductions recovers 0.44 of the 0.51 drop, while survivor weighting and the gradient estimator together leave a remainder that four seeds do not resolve. Measured at evaluation time instead of training time, the cost appears two and a half times larger, which is why sufficiency must be established by retraining.

What the selection does. When the budget exceeds the objects a labeled selector rates confidently, it fills the leftover slots with low-scored objects, which costs about a fifth of the success on CALVIN. What improves success is less the selection the explanation points to than the content of the annotation list, which beats a permuted list, and on RoboCasa a smaller graph, which helps regardless of which objects it keeps. A selection that looks beneficial against the full input can therefore look ordinary against a random subset of the same size.

Benchmarks decide what can be shown. CALVIN leaves room for the scene, with a blind policy far below the full one, whereas on RoboCasa a robot-only policy already exceeds the full graph, and under distractors on RoboTwin every policy trained on clean scenes falls to the level of a blind policy. Claims that a selected subset explains a policy should therefore be made against floors and matched random baselines on the benchmark in question, not against the full input.

6 Conclusion

We stated three conditions for increasingly strong readings of the kept set under which the subset selected by an In-Loop Subset Explanation explains the policy that selected it, Discovery, Removal and Consequence, together with a test for each. Applied to object-centric policies on three manipulation benchmarks and to three external methods, the tests show that excluded objects leak into the action through standard operators, that budgets fill leftover slots with low-scored objects, and that no tested design point satisfies all three conditions. Labeled selectors satisfy Removal and Consequence but not Discovery, label-free selectors satisfy Discovery and Removal but reach at most the level of random selection, and closing the leaks costs up to a quarter of performance. A selection signal that satisfies all three conditions at once remains open.

Limitations. Our certificates hold given the selection, so they do not rule out that the choice of objects itself carries information about the excluded ones, which our Discovery test checks only for the annotation list. Robot links are selectable, as in SIR, so our Consequence verdicts partly measure whether proprioception is kept, and the three conditions are necessary for their claims but not jointly sufficient. Our tests cover one policy family in simulation and one label-free estimator, whose budget uses a constant derived from the labeled selector's kept fraction, and the Consequence verdicts rest mainly on CALVIN. On CALVIN most of the removal cost is through the reductions over excluded objects, but the remainder from survivor weighting and the gradient estimator is not resolved at four seeds.

Future work. The most direct next step is a label-free selection signal that beats a random subset of the same size, from other estimators or from task-related signals such as object motion, together with tests for information carried by the choice of objects itself. Benchmarks should report robot-only, blind and matched random floors and include distractors in training, so that a selection has something to decide.

AI Use Statement

In this work, we used generative AI tools (Anthropic's Claude models) for implementing methods, providing feedback on the research methodology and experiments, supporting the analysis of experimental data, and discussion of results. We have not used generative AI tools for conceptual frameworks, proposing or refining hypothesis, generating synthetic data sets, cleaning or reformatting datasets, formulating mathematical claims, or providing ingredients for or writing proofs, and translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for writing and debugging code, including the code for figures, searching

the literature, and drafting and editing the manuscript. We have reviewed all AI-assisted work: The authors designed the study, fixed every prediction and decision rule before the corresponding evaluation, and verified every reported number, citation and claim. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Appendix A.1 describes the benchmarks, objects, architecture, hyperparameters, design points and how every random baseline is drawn, together with the pre-registered predictions, and Appendices A.2 to A.7 report all results, contrasts, tests and quality checks. The code, trained checkpoints and evaluation records will be released after the review process.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

Kenza Amara, Jovan Tanevski, Rex Ying, and Ce Zhang. GInX-Eval: Towards in-distribution evaluation of graph neural network explanations. *arXiv preprint arXiv:2309.16223*, 2023.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl3m1BFDB.

Steve Azzolin, Antonio Longa, Stefano Teso, and Andrea Passerini. Perks and pitfalls of faithfulness in regular, self-explainable and domain invariant gnns. *arXiv preprint arXiv:2406.15156*, 2024.

Steve Azzolin, Stefano Teso, Bruno Lepri, Andrea Passerini, and Sagar Malhotra. GNN explanations that do not explain and how to find them. In *International Conference on Learning Representations (ICLR)*, 2026.

Rokas Bendikas, Daniel Dijkman, Markus Peschl, Sanjay Haresh, and Pietro Mazzaglia. Focusing on what matters: Object-agent-centric tokenization for vision language action models. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=Ict1OjU9gl.

Alexandre Chapin, Bruno Machado, Emmanuel Dellandrea, and Liming Chen. Object-centric representations improve policy generalization in robot manipulation. *arXiv preprint arXiv:2505.11563*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Yongqiang Chen, Yatao Bian, Bo Han, and James Cheng. How interpretable are interpretable graph neural networks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6413–6456. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chen24b.html.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.

Coline Devin, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Deep object-centric representations for generalizable robot learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7111–7118. IEEE, 2018.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4443–4458, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.408. URL https://aclanthology.org/2020.acl-main.408/.

Lukas Faber, Amin K. Moghaddam, and Roger Wattenhofer. When comparing to ground truth is wrong: On evaluating gnn explanation methods. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 332–341, 2021.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Peter Hase, Harry Xie, and Mohit Bansal. The out-of-distribution problem in explainability and search methods for feature importance explanations. *Advances in neural information processing systems*, 34:3650–3666, 2021.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. *Advances in Neural Information Processing Systems*, 35:23386–23397, 2022.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4198–4205, 2020.

Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C Wallace. Learning to faithfully rationalize by construction. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4459–4473, 2020.

Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *International Conference on Artificial Intelligence and Statistics*, pp. 1459–1467. PMLR, 2021.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. Pmlr, 2020.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. pmlr, 2019.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 107–117, 2016.

Yixin Lin, Austin S Wang, Eric Undersander, and Akshara Rai. Efficient and interpretable robot manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):2740–2747, 2022.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Paul Mattes, Jan Schwab, Jens Bosch, Maximilian Xiling Li, Nils Blank, Minh-Trung Tang, Moritz Haberland, and Rudolf Lioutikov. Sir: Structured image representations for explainable robot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 42484–42493, June 2026.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. *Advances in neural information processing systems*, 33:2220–2231, 2020.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International conference on machine learning*, pp. 15524–15543. PMLR, 2022.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. SOLD: Slot object-centric latent dynamics models for relational manipulation learning from pixels. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44911–44935. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/mosbach25a.html.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. *Advances in neural information processing systems*, 32, 2019.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

Sagar Parekh, Preston Culbertson, and Dylan P Losey. Transmask: Masked state representation through learned transformation. *arXiv preprint arXiv:2603.05670*, 2026.

Jongjin Park, Younggyo Seo, Chang Liu, Li Zhao, Tao Qin, Jinwoo Shin, and Tie-Yan Liu. Object-aware regularization for addressing causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 34:3029–3042, 2021.

Aahlad Puli, Nhi Nguyen, and Rajesh Ranganath. Explanations that reveal all through the definition of encoding. *Advances in Neural Information Processing Systems*, 37:99965–100006, 2024.

Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgzLkBKPB.

Han Qi, Changhe Chen, and Heng Yang. Compose by focus: Scene graph-based atomic skills. *arXiv preprint arXiv:2509.16053*, 2025.

Jianing Qian, Qinhe Peng, Emmanuel Panov, Leonor Fermoselle, Dinesh Jayaraman, Bernadette Bucher, and Tarik Kelestemur. Expanding spatial and temporal context for robotic imitation learning with scene graphs. *arXiv preprint arXiv:2606.01072*, 2026.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, A. Sophia Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 459–470. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/renz23a.html.

Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *Robotics: Science and Systems*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Giuseppe Serra and Mathias Niepert. L2xgnn: learning to explain graph neural networks. *Machine learning*, 113(9):6787–6809, 2024.

Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11962–11971, 2021.

Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pp. 414–424, 2020.

Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion. In *International Conference on Learning Representations*, volume 2025, pp. 35011–35030, 2025.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Junchi Yu, Jie Cao, and Ran He. Improving subgraph recognition with variational graph information bottleneck. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19374–19383. IEEE, 2022.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi Jaakkola. Rethinking cooperative rationalization: Introspective extraction and complement control. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4094–4103, 2019.

Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3418–3433. PMLR, 06–09 Nov 2023a. URL `https://proceedings.mlr.press/v229/zhu23b.html`.

Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, pp. 1199–1210. PMLR, 2023b.

A Appendix

A.1 Implementation Details

Benchmarks and objects. On RoboCasa we evaluate 200 rollouts per task, one per scene, on all 24 atomic tasks, with the 16 tasks outside pick-and-place as the primary result, on a reseeded evaluation sampler, and values from an earlier sampler are labeled legacy (Appendix A.7). On CALVIN we follow the official protocol with 1,000 chains of five instructions per seed, and on RoboTwin 2.0 we evaluate four tasks (bowls, container, hammer and bread) with 50 rollouts per task and seed, on clean scenes and with added distractors. Seeds are 42, 43, 44 and 101010 on CALVIN and RoboCasa, and 101010, 43 and 7 on RoboTwin. Objects and their features are extracted from the static cameras following SIR (Mattes et al., 2026), and on RoboCasa each node carries a crop-encoder appearance

feature and normalized bounding-box coordinates in two views, with nodes fully connected within a frame. Every node is a robot link, a movable object or a fixture, and the robot links are the policy's only proprioceptive input. The object annotation list marks the relevant nodes per task and always includes robot links, the mobile base and gripper on RoboCasa and the gripper links on CALVIN, which gives 3.38 relevant nodes per task on the 16 RoboCasa tasks and 6, 5, 4 and 4 on the four RoboTwin tasks. A validation frame contains 12.3 nodes on RoboCasa and 15.7 on CALVIN, of which 7.35 are robot links and 1.37 annotated scene objects.

Architecture and hyperparameters. All policies use the MDT denoiser (Reuss et al., 2024) without its auxiliary objectives, trained with AdamW at a learning rate of 10^{-4} and a weight decay of 10^{-3}, with a 5% warmup and a step decay by half every 10% of training. On RoboCasa, the denoiser has an embedding size of 512, four encoder and six decoder layers with 16 heads, at most 32 graph tokens and an action window of 10, and samples with 10 DDIM steps. The graph encoder is a two-layer graph attention network with hidden size 512 and four heads, and the selector a two-layer transformer of the same size, conditioned on the instruction. Table 3 lists the settings that differ across benchmarks, and all policies within a benchmark are trained for the same number of optimizer steps, except MGS (Appendix A.6).

Table 3: Training settings that differ across benchmarks. VGIB's noise term has weight 0.01.

	CALVIN D→D	CALVIN ABC→D	RoboCasa	RoboTwin
Batch size / epochs	128 / 50	128 / 28	128 / 50	16 / 50
Training frames	293,378	1,017,146	337,491	23,523
Optimizer steps	114,600	222,488	131,850	73,500
Loss weight, annotation list	0.1	0.1	0.1	0.1
Loss weight, sparsity (label-free)	0.01	–	–	0.1
Loss weight, score spread	0.1	0.1	0.1	0.1
Target fraction r (label-free)	0.2788	–	–	0.598
Threshold, VGIB	0.6	0.6	0.6	–

Design points and random baselines. Table 4 lists the selection rule, the removal path and the kept fraction of every design point. The label-free design point keeps $\lceil r\, n_t \rceil$ nodes, so its realized kept fraction lies above r. Random baselines differ in how they are drawn. On CALVIN, the random baselines train with a mask resampled at every step, like the label-free design points, and evaluate a random ranking fixed per task, visible node set and seed. On RoboTwin, both random baselines use one fixed random ranking per task, node set and seed in training and evaluation, which gives them a more consistent input than the sampled training mask of the label-free design point. On RoboCasa, the random list is drawn once per task and seed, keeps every robot node, and matches the size of the fixed annotation list.

Table 4: Selection rule, removal path and kept fraction on validation frames. On RoboTwin the leaky selector's readout lowers its effective kept fraction to 0.55–0.56.

Design point	Rule	Path	CALVIN	RoboCasa	RoboTwin
Labeled, leaky (SIR)	top-k, k from list	leaky	0.279	0.2905	0.59–0.60
Labeled, corrected	top-k, k from list	corrected	0.279	0.2905	0.60
Labeled, leaky reductions only	top-k, k from list	corrected except R2	0.279	–	–
Label-free, leaky (VGIB)	threshold 0.6	noise	0.688	0.466	–
Label-free (ours)	top $\lceil rn_t \rceil$	corrected	0.311	–	0.645–0.650
Label-free, robot links kept	robot links, top $\lceil rn_{\text{scene}} \rceil$	corrected	0.65	–	–
Random selection	matched count	corrected	0.311	4.44 nodes	0.60 / 0.65
Random, robot links kept	robot links, random scene objects	corrected	0.65	–	–

Statistics and pre-registration. Differences are paired by seed with t-based 95% intervals over seeds, minimum detectable effects (MDE) for a two-sided test at 5% with 80% power, and on RoboTwin a Fisher test on pooled rollouts as a secondary test. Before evaluating a primary comparison we committed a prediction and a reading rule to a timestamped record, and Table 5 lists the outcomes.

Table 5: Pre-registered predictions and outcomes, in average completed instructions on CALVIN, percent on RoboCasa and rates on RoboTwin.

Benchmark	Quantity	Predicted	Observed	Outcome
RoboCasa	corrected − leaky, 16 tasks	within ± 2	-3.46	missed
RoboCasa	soft weighting with the list, share of the leaky margin	$\geq 80\%$	42%	not met
RoboCasa	permuted list and no list loss, vs unweighted graph	within ± 2.4	-2.07, $+0.84$	met
RoboCasa	annotation list − random list, legacy, 24 tasks	in $(0, +1.5)$	$+0.95$	met
RoboCasa	pixel policy, reseeded vs legacy	within ± 1.5	12.84 vs 12.47	met
CALVIN	label-free design point	[1.00, 1.75]	0.46	missed
CALVIN	random selection	[0.75, 1.35]	0.70	missed
CALVIN	label-free, robot links kept	[0.80, 1.30]	1.18	met
CALVIN	random, robot links kept	[0.90, 1.40]	1.12	met
CALVIN	label-free − random, robot links kept	$[-0.30, +0.10]$	$+0.06$	met
CALVIN	labeled, leaky reductions only	[1.50, 2.20]	2.06	met
CALVIN	leaky reductions only − corrected	$[-0.10, +0.60]$	$+0.44$	met
CALVIN	leaky − leaky reductions only	$[-0.05, +0.60]$	$+0.06$	met
RoboTwin	label-free design point	0.20	0.130	missed
RoboTwin	label-free − random	$+0.03$	-0.092	missed

A.2 Full Results

Main results and contrasts. Table 6 extends Table 2 with standard deviations over seeds and the variants of the appendix, Table 7 lists the paired contrasts, and Table 8 the per-seed values on CALVIN. On CALVIN, the labeled selector beats random selection in every seed, by 0.84 to 1.05, although the random baseline keeps slightly more objects per frame (4.88 against 4.38), and the label-free design point falls below it in every seed. On RoboCasa the annotation list beats the random list in every seed, by 0.25 to 1.59 points. On RoboTwin's clean scenes, the corrected labeled selector reaches 23.0, 26.0 and 28.5% on seeds 101010, 43 and 7, its random baseline 8.5, 18.0 and 21.0%, the label-free design point 12.0, 12.0 and 15.0%, and its random baseline 17.0, 22.0 and 27.5%.

Table 6: Success with standard deviations over seeds, in average completed instructions on CALVIN and percent on RoboCasa and RoboTwin. RoboTwin uses three seeds and the blind policy two. RoboTwin's leaky selector counts the first of two evaluations of seed 7, as fixed before either was read, and for the pixel policy seed 101010 is the mean of three evaluations and seed 43 ran 400 rollouts.

Design point	D→D	ABC→D	RoboCasa	RoboTwin clean	RoboTwin distr.
Pixel policy	1.76 (0.04)	1.40 (0.11)	12.84 (0.29)	46.25 (1.95)	1.50 (0.50)
Blind policy	0.44 (0.06)	–	–	1.50 (0.71)	2.50 (0.00)
Robot-only	1.08 (0.09)	–	19.85 (0.82)	–	–
Unsparsified graph	1.25 (0.05)	1.04 (0.05)	17.37 (0.69)	22.17 (1.15)	0.33 (0.58)
Fixed annotation list	–	–	20.09 (0.70)	–	–
Random, labeled size	0.70 (0.03)	–	19.01 (0.13)	15.83 (6.53)	–
Random, label-free size	0.70 (0.03)	–	–	22.17 (5.25)	1.00 (0.87)
Random, robot links kept	1.12 (0.09)	–	–	–	–
Labeled, leaky	2.12 (0.15)	2.20 (0.08)	25.09 (0.60)	26.67 (1.15)	0.83 (1.04)
Labeled, corrected	1.62 (0.07)	1.71 (0.04)	21.62 (0.82)	25.83 (2.75)	–
Labeled, leaky reductions only	2.06 (0.07)	–	–	–	–
Label-free, leaky	1.14 (0.11)	1.11 (0.09)	18.55 (0.99)	–	–
Label-free (ours)	0.46 (0.07)	–	–	13.00 (1.73)	3.33 (1.44)
Label-free, robot links kept	1.18 (0.03)	–	–	–	–

RoboCasa on all 24 tasks. Table 9 lists success on all 24 RoboCasa tasks, including pick-and-place, where the corrected selector falls 2.31 points [1.46, 3.17] below the leaky one, and arms evaluated on the 16 tasks only have no value.

Table 7: Paired contrasts over seeds, in average completed instructions on CALVIN and percentage points elsewhere, with – where the MDE was not computed. On RoboTwin the rollout-level Fisher test gives $p = 2.6 \times 10^{-5}$ for labeled against random and $p = 3.9 \times 10^{-5}$ for label-free against random at three seeds, the six-seed extension was declared after an interim look, and over nine seeds the corrected policy's relative cost lies in $[-16.2\%, +5.9\%]$. The RoboCasa contrast of the corrected selector with the random list is not matched in size, and the labeled selector that keeps every robot link uses a per-task budget.

Benchmark	Contrast	Difference	95% interval	MDE
CALVIN D→D	corrected − leaky	−0.506	[−0.665, −0.346]	–
CALVIN ABC→D	corrected − leaky	−0.490	[−0.608, −0.373]	–
RoboCasa	corrected − leaky	−3.46	[−4.74, −2.18]	1.67
RoboTwin	corrected − leaky, 3 seeds	−0.8	[−10.3, +8.7]	–
RoboTwin	corrected − leaky, 9 seeds	−1.4	[−4.4, +1.6]	4.2
CALVIN D→D	leaky reductions only − corrected	+0.443	[+0.303, +0.584]	0.183
CALVIN D→D	leaky − leaky reductions only	+0.062	[−0.121, +0.246]	0.240
CALVIN D→D	corrected − pixel	−0.145	[−0.222, −0.068]	–
CALVIN D→D	labeled − random	+0.920	[+0.775, +1.064]	0.189
RoboTwin	labeled − random	+10.0	[+0.3, +19.7]	12.1
RoboCasa	annotation list − random list	+1.09	[+0.12, +2.05]	1.26
RoboCasa	corrected − random list	+2.62	[+1.47, +3.77]	1.50
RoboCasa	corrected − robot-only	+1.77	[+0.19, +3.35]	2.07
CALVIN D→D	label-free − random	−0.239	[−0.343, −0.135]	–
RoboTwin	label-free − random, 3 seeds	−9.2	[−18.7, +0.3]	11.8
RoboTwin	label-free − random, 6 seeds	−4.3	[−11.9, +3.2]	10.2
CALVIN D→D	label-free − random, robot links kept	+0.056	[−0.121, +0.233]	0.232
CALVIN D→D	labeled − random, robot links kept	+0.177	[−0.050, +0.404]	0.297
CALVIN D→D	annotation's budget − label-free, at evaluation	−0.023	[−0.055, +0.010]	0.043

Table 8: Per-seed success on CALVIN D→D, as the average number of completed instructions over 1,000 chains.

Design point	Seed 42	Seed 43	Seed 44	Seed 101010	Mean (SD)
Labeled, leaky	2.327	2.137	2.010	2.011	2.121 (0.150)
Labeled, corrected	1.718	1.566	1.570	1.609	1.616 (0.071)
Labeled, leaky reductions only	2.092	2.137	2.000	2.007	2.059 (0.067)
Random selection	0.669	0.729	0.679	0.708	0.696 (0.027)
Label-free (ours)	0.494	0.525	0.354	0.456	0.457 (0.074)
Label-free, robot links kept	1.217	1.144	1.179	1.172	1.178 (0.030)
Random, robot links kept	0.999	1.105	1.194	1.191	1.122 (0.092)

All evaluated methods, larger pixel budgets and policies without a graph encoder. Table 10 lists every method we evaluated on the 16 RoboCasa tasks, where Table 2 reports the reseeded method of each category. With a per-object MLP encoder instead of message passing, the policy reaches 20.69%, above the unsparsified graph attention network (17.30%), and mean pooling lowers every variant except VGIB. On CALVIN and RoboTwin each category has one method, except for a pixel policy trained from scratch on RoboTwin (23.8% against 46.3% for the pretrained one), and a mean-pooled variant of the leaky labeled selector on RoboTwin reaches 19.0% on one seed.

A.3 Discovery

Permutation test. We replace the object annotation list by a permuted list and by a list that raises an error when read, and compare actions, kept sets and training loss. The label-free design points and their random baselines are bit-identical in training and evaluation mode on CALVIN and in all eight combinations on RoboTwin, whereas the labeled selector's training loss changes on both, by 9.6×10^{-3} on CALVIN, and on RoboCasa its actions differ by up to 2.03. The tests run in double precision on CPU with trained weights and certify the code path, including every channel through

Table 9: Success in percent on all 24 RoboCasa tasks, with six seeds for the legacy annotation list and random list.

Design point	Reseeded	Legacy
Labeled, leaky	16.76	16.91
Labeled, corrected	14.45	–
Label-free, leaky (VGIB)	–	12.43
Robot-only	13.27	–
Unsparsified graph	11.62	11.59
Fixed annotation list / random list	–	13.83 / 12.88
Soft weighting, true / permuted / no list	13.85 / 10.26 / 12.19	–

Table 10: Every method evaluated on the 16 RoboCasa tasks, in percent. Legacy and reseeded values use different samplers and are not compared with each other, and the legacy random and annotation lists use six seeds. MGS is reported in Appendix A.6.

Category	Method	Legacy	Reseeded
Labeled, leaky	top-k, per-node readout / mean pooling	25.23 / 23.52	25.09
Labeled, corrected	top-k, true removal	–	21.62
Unsparsified graph	graph attention, per-node readout / mean pooling	17.30 / 19.51	17.37
Per-object MLP, no selection	per-node readout / mean pooling	20.69 / 17.59	–
Label-free, leaky	VGIB, graph, per-node readout	18.54	18.55
	VGIB, graph, mean pooling	21.02	–
	VGIB, per-object MLP, per-node readout / mean pooling	20.16 / 22.56	–
Robot-only	true removal	–	19.85
Random selection	random list, size of the annotation list	19.26	19.01
Annotation list	fixed list, true removal	20.66	20.09

which the list could set the budget, but they cannot detect a constant derived from the list and fixed in the configuration, such as the label-free target fraction r.

Agreement with the annotation list. Table 11 lists the AUROC of each selector's scores against the annotation list on CALVIN. Because the list marks robot links as relevant, a selector that drops robot links scores below chance on all nodes, independently of what it keeps from the scene. On RoboCasa, where scores are not stored, the leaky and the corrected labeled selector reach a precision of 0.952 and 0.949 and a recall of 0.981 and 0.978 against the list at rollout, and VGIB 0.565 and 0.715.

Table 11: AUROC against the annotation list on CALVIN validation frames, as means over four seeds.

Selector	All nodes	Scene nodes
Labeled, leaky	0.9997	0.9995
Labeled, corrected	0.9982	0.998
Label-free, leaky (VGIB)	0.696	0.634
Label-free (ours)	0.345	0.436
Label-free, robot links kept	–	0.268

What the annotation list contributes. Weighting all objects softly on RoboCasa, without removing any, reaches 20.64% (SD 0.40) with the true list, 15.30% (0.57) with a permuted list and 18.21% (1.39) without a loss on the list, against 17.37% (0.69) unweighted (Table 9 gives the 24-task values). The true list exceeds the permuted one by 5.34 points [3.83, 6.86] and training without the list by 2.43 points [-0.13, 4.99], and the permuted list falls 2.07 points [0.49, 3.65] below the unweighted graph.

Non-decision diagnostics. We call an object confidently scored if its score is at least $\tau = 0.99$, with all results unchanged for τ between 0.90 and 0.999, and a frame is a no-choice frame if the budget keeps every confidently scored object. The gap at the cut is the difference between the last kept and the first dropped score, normalized by the range of scores in the frame, and all objects, including robot links and fixtures, enter the ranking. Table 12 summarizes the diagnostics on validation frames for CALVIN and RoboCasa and on clean rollout frames for RoboTwin. With a budget from the annotation list, the labeled selector saturates its scores, and the budget typically equals the number of saturated objects on CALVIN and RoboTwin (74% and 68% of frames) and exceeds it by 1.6 to 1.7 objects on RoboCasa, where the remaining slots go to objects rated below 0.5. The label-free leaky selector's threshold of 0.6 keeps every object in 21.1% of CALVIN frames (7.3–34.7% across seeds), while the label-free design point does not saturate, since its scores are pulled toward r. Under distractors on RoboTwin the budget binds in every frame and the cut falls inside a tie in 22.5–30.2% of frames, where CPU and GPU keep different sets in 182 of 307 tied frames.

Table 12: Non-decision diagnostics in percent of frames, except the median gap, with ranges over seeds. The median gap of the labeled selectors is computed over no-choice frames, RoboCasa gaps and ties use all 24 tasks, and the RoboTwin labeled row is measured on the second fix. – not defined for the rule.

Benchmark	Design point (rule)	No-choice	Kept below 0.5	Median gap	Gap < 0.01	Exact ties
CALVIN	Labeled, leaky (top-k)	99.97	2.6	0.83	2.0	0.00
CALVIN	Labeled, corrected (top-k)	99.94	4.0	0.84	3.7	0.00
CALVIN	Label-free, leaky (threshold 0.6)	–	–	0.11–0.15	0.8	–
CALVIN	Label-free (top $\lceil rn_t \rceil$)	–	–	0.07	9.7	–
RoboCasa	Labeled, leaky (top-k)	100	11.0	0.82	8.0–10.5	0.00–0.01
RoboCasa	Labeled, corrected (top-k)	100	11.3	0.82	9.9–16.6	0.18–0.45
RoboTwin	Labeled, leaky (top-k)	97.3	27.9	0.84	19–31	0.00
RoboTwin	Labeled, second fix (top-k)	93.8	30.6	0.85	19–31	0.00
RoboTwin	Label-free (top $\lceil rn_t \rceil$)	–	1.4–3.4	0.14–0.16	4.6–8.0	0.00

Budget interventions. Table 13 changes the labeled selector's budget on CALVIN while keeping everything else fixed. Adding two slots per frame, filled with low-scored objects, lowers the corrected selector from 1.616 to 1.290 and raises the leaky one from 2.121 to 2.194, while supervising the added slots toward the robot links raises the corrected selector by 0.385, to 1.674. Giving the label-free design point the annotation's per-task budget at evaluation lowers its success from 0.457 to 0.435 (Table 7), which redistributes the count across tasks at an almost unchanged total and so does not measure training with a budget from the list.

Table 13: Budget interventions on CALVIN D→D, four seeds, average completed instructions.

Budget	Kept per frame	Leaky	Corrected	Robot links kept
k from the list	4.38	2.121	1.616	0.41
$k + 2$	6.38	2.194	1.290	0.53
$k + 2$, extra slots supervised to robot	6.38	1.875	1.674	0.68
≈ 10.35, all robot links supervised	10.35	1.882	1.299	1.00
≈ 10.35, other labels	10.35	1.163	1.050	0.48

What the label-free selector keeps. Table 14 compares the label-free design points on CALVIN with their random baselines and the labeled selector. The kept sets of the label-free design point agree across the six pairs of training seeds with a mean Jaccard overlap of 0.43, against 0.20 for random selection and 0.99 for the labeled selector, yet consecutive kept sets differ by a Jaccard distance of 0.18, against 0.007 for the labeled selector and 0.036 for random selection, and 0.13 against 0.05 on RoboTwin rollouts, while the visible objects stay the same. Its scores are spread rather than collapsed onto r, and at the final checkpoint the gradient of the policy loss on the scores is 15 to 25 times larger than that of either regularizer. Per task, it keeps 7 to 21% of the robot links against 38 to 45% for the labeled selector and falls below the labeled selector on 32 of 33 tasks.

With every robot link kept, its scene ranking keeps 5% of the annotated scene objects against 40% of the others, while the matching random baseline keeps both at about a third.

Table 14: Retention by node category on CALVIN validation frames, as the fraction of visible nodes of each category that is kept.

Design point	Kept per frame	Robot links	Annotated scene	Unannotated scene
Labeled, corrected	4.38	0.41	1.00	0.01
Label-free (ours)	4.88	0.14	0.40	0.47
Random selection	4.88	0.32	0.33	0.30
Label-free, robot links kept	10.19	1.00	0.05	0.40
Random, robot links kept	10.19	1.00	0.32	0.34

A.4 REMOVAL

The construction test. We build $\mathcal{C}(S_t)$ by slicing the graph to the kept nodes and the edges between them, recomputing every normalization statistic on the kept set, and reusing the kept nodes' raw features, and compare it with the masked full input at the encoder output or the action, with identical denoiser noise. A design point passes if the residual stays at the numerical floor, measured between identical passes and reorderings of the input, and if a control of the same size that differs only in which nodes are masked is bit-identical, while a configuration built to leave one route open must fail. Table 15 lists the certificates.

Table 15: Construction-test certificates on trained weights. The built-to-fail cases are the leaky mechanics on the same weights or an injected score-weighted readout.

Benchmark	Design point	Point	Residual	Built to fail
CALVIN	Labeled, corrected	encoder	0.0 (floor $\leq 2 \times 10^{-15}$)	0.79–0.91
CALVIN	Label-free design points and their baselines	action	at floor	–
RoboCasa	Labeled, corrected	encoder	9.5×10^{-7} (float32)	1.42–2.84
RoboTwin	Labeled, corrected	action	bit-identical (float64)	–
RoboTwin	Label-free (ours) and its baseline	action	$2.7\text{–}5.8 \times 10^{-15}$	2.1–2.3

Leak sites and their size. Table 16 lists where excluded objects re-enter each leaky design point, and the correction closes every listed site of the labeled selector. To size the leak, we replace the features of excluded objects in the leaky CALVIN policy with features of the same type from other frames, keeping the kept set fixed, over 384 validation frames per seed and eight noise draws. The action changes by 0.113 (SD 0.004) of the policy's own sampling spread and by up to 2.11 on single frames, which reverses the gripper command, fixing the scores as well leaves this at 0.113, and the corrected policy changes by exactly zero.

Table 16: Leak sites by class. Scores computed while reading every object make score weighting both survivor distortion and value-dependent. The count of kept objects under a threshold is part of the selection and so lies outside R-C, which holds given the selection.

Site	Class	Present in	Closed
Softmax denominator of message passing	value-dependent	leaky SIR, leaky reductions only, VGIB	yes
Edge weights gated after the softmax	value-dependent	leaky SIR (CALVIN, RoboCasa)	yes
Graph-level normalization over all nodes	value-dependent, presence	leaky SIR, leaky reductions only (CALVIN)	yes
Readout weighted by scores from all objects	survivor distortion	leaky SIR, soft weighting	yes
Kept nodes scaled by their score at the graph input	survivor distortion	first RoboTwin fix	yes
Noise substitute taken over all nodes	value-dependent	VGIB	no
Count of kept objects under a threshold	selection	VGIB, MGS	–

External methods. Table 17 applies the construction test to the three external method families, top-k and self-attention graph pooling, a self-explaining graph network (GSAT) and differentiable

token pruning (DynamicViT), with version and checkpoints fixed before the test. A method passes R-C if nothing from outside the retained set reaches its output, and we report separately whether it reweights the retained objects. Default top-k pooling passes, since its gate depends only on each kept node's own features, but reweights survivors, and with an exact indicator in place of this gate its scorer receives no gradient in 16 of 16 runs. The self-explaining graph network never commits to a subset: With the top half of edges as the retained set, the excluded edges keep a median attention of 0.45 to 0.59, although its explanations reach an AUROC of 0.98 against ground truth. Differentiable token pruning passes on its deployed inference path, bitwise on 190 of 190 images, given its retained sets and the hidden states entering the first pruning block, while its masked training path fails by about 10^{-6} through a smoothing constant. Closing only one route was worse than closing none in every method, for example raising the median relative residual of top-k pooling from 0.25–0.30 to 0.38–0.74.

Table 17: Construction test of external methods, as the largest absolute residual at the pooling output or the logits, with ranges over seeds, datasets and definitions of the retained set. Pooling operators are trained on PROTEINS and MUTAG, GSAT on BA-2Motifs, and DynamicViT is the published DeiT-S checkpoint with keep ratio 0.7.

Method	Excluded-set dependence	Survivor reweighting	R-C
Top-k pooling, default	0.0	1.9–7.6	pass
Top-k pooling, minimum score	0.007–3.0	2.0–11.2	fail
Self-attention pooling (both)	1.3–7.6	2.1–9.2	fail
GSAT, GIN and PNA backbones	5.3–12.8	5.0–11.8	fail
DynamicViT, inference	0.0	0.0	pass
DynamicViT, training path	1.1×10^{-6}	1.3×10^{-6}	fail

Partial fixes, gradients and the source of the removal cost. On RoboTwin the leaky policy reaches 26.7% over three seeds, a first fix that scaled kept objects by their scores at the graph input 21.5%, a second fix that kept a score-weighted readout 25.2%, and the final fix, whose 79,488 recorded readout weights all equal one, 25.8%. Over nine seeds, the final fix reaches 25.7% against 27.1% for the leaky policy, a relative cost of at most 16% (Table 7). Under true removal, excluded objects receive exactly zero policy gradient in our parameterization, which is why the label-free design point samples its training mask, whereas on the leaky path they receive about ten times more gradient than kept ones in one measured batch. To separate the reductions from the other changes, a CALVIN variant of the corrected policy reverses only R2, so that reductions again run over all objects while survivors keep weight one and the scorer trains through the straight-through estimator. It reaches 2.06, 0.44 [0.30, 0.58] above the corrected policy and 0.06 $[-0.12, +0.25]$ below the leaky one, so reversing R2 alone recovers 88% of the drop from 2.12 to 1.62, and survivor weighting and the gradient estimator together account for a remainder that four seeds do not resolve. Because reversing R2 also changes how many objects each reduction normalizes over, this recovery bounds the value of leaked information from above rather than isolating it.

A.5 Consequence

Distractors and evaluation-time removal. Under distractors, the pixel policy reaches 1.5%, SIR 0.8% and the unsparsified graph 0.3%, against 2.5% for the blind policy, and the label-free design point 3.3% against 1.0% for its random baseline, all at or near the blind level (Table 6). A two-by-two design that crosses training data, clean or with distractors, with the encoder places the failure in the training data ($p = 10^{-8}$) rather than the encoder ($p = 0.84$), and 61% of the objects SIR keeps under clutter are distractors. Closing the leak only at evaluation, on the trained leaky RoboCasa checkpoints, lowers success from 25.09% to 16.27%, a loss of 8.81 points [6.44, 11.19], against 3.46 when the policy is retrained with true removal, since the evaluation-time operator changes the leaky encoder's output by 31 to 34%. Removing single objects at evaluation time changes success in none of 70 tests after correction for multiple comparisons.

Extensions. On RoboTwin we extended the label-free contrast to six seeds after an interim look at two seeds, triggered by additional compute, and report it beside the pre-registered three-seed result. Over six seeds the label-free design point reaches 14.7% against 19.0% for its random baseline, 4.3

points below $[-11.9, +3.2]$, with two of six seeds reversing (rollout-level Fisher $p = 0.005$). On CALVIN we added, after the main results, a label-free design point that keeps every robot link and selects among scene objects only, against a random baseline that does the same. It reaches 1.18 against 1.12 (+0.06 $[-0.12, +0.23]$), up from 0.46, so its deficit came mainly from dropping robot links, and with every robot link kept the labeled selector does not detectably beat random scene selection either (+0.18 $[-0.05, +0.40]$, Table 7). Its explanation covers scene objects only, and it shares one annotation-derived constant and the count of kept objects with its baseline.

A.6 Motion-Grounded Selection

MGS selects from labels derived from which objects move in the demonstrations, not from the object annotation list. Its selector, a two-layer transformer over all objects conditioned on the instruction, is trained first and frozen, and the policy is then trained on its kept set, a select-then-predict design (Jain et al., 2020) whose selection the policy cannot shape (Jethani et al., 2021). It keeps every object scoring above 0.5, always keeps three robot links, keeps an object once detected for the rest of the episode, and runs on the corrected path. MGS was trained on the 16 evaluated RoboCasa tasks only, for 81,450 optimizer steps (62% of the shared budget), and evaluated on the first 100 scenes. It passes the permutation test, its readout weights are exactly one, and only 0.26% of its scores lie within 0.05 of the cut. It reaches 20.44% against 18.80% for a random selection of its size, +1.64 points $[-1.74, +5.02]$ on 16 tasks and +2.44 $[-0.11, +4.99]$ on the 12 tasks where it keeps any object, and against 22.02% for the labeled selector on the same scenes. Because its selector attends across objects, the number of objects above the cut depends on the excluded objects' values, a channel that lies in the selection and so outside R-C.

A.7 Quality Control

On RoboCasa, a failed renderer can complete an evaluation and score every rollout as a failure, so every unit of one task and seed is compared against its sibling seeds and flagged units are rerun on another machine, which found at least 108 of 4,592 corrupt units (2.35%, a lower bound). The reseeded sampler makes the denoiser's starting noise a deterministic function of episode and step, lowering disagreement between repeated evaluations from 22.8% to 1.2% of scenes, and no comparison crosses samplers. On RoboTwin, renderer out-of-memory errors went to a log that earlier checks did not read, and all five affected evaluations were repeated from the same checkpoints, replacing the originals whatever their value, while on CALVIN two evaluations on a faulty machine and a builder that silently dropped excluded machines under load were found and fixed. During the project we also found and corrected, before any affected value entered the paper, a graph library that mutated batches in place, survivors left scaled by their scores in an early top-k implementation, a result scored before three corrupt cells were repaired, minimum detectable effects computed with a normal approximation, a kept fraction estimated from the wrong model, evaluations converted to another sampler, an aggregation that kept only the last seed, a double-counted checkpoint, an untraceable leak figure, an invalid reordering floor, a gate test with a bound below chance, a pixel value from a run with batch size 16, and several transcription and file-naming errors in the external tests, none of which changed a verdict.